Parsing

Lizhen Q

Overview of the NLP Lectures

- Introduction to natural language processing (NLP).
- Regular expressions, sentence splitting, tokenization, part-of-speech tagging.

- Language mod

- Vector semantics.

- Parsing.
  - Dependency parsing.
- Semantics.

Relation Extraction

- Find *worksFor*(*entity_a*, *entity_b*) relation from text.

Mark has worked for Telstra.

Per Org

training data

Per *worksFor* Org

John works for Facebook.

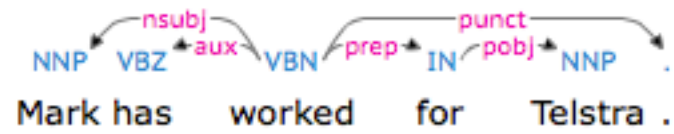
Per Org

Mark Zuckerberg said he would have worked for Microsoft.

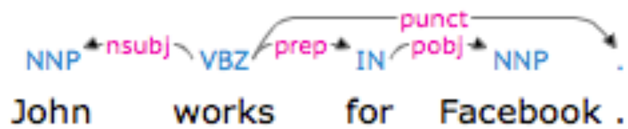
Per Org Org

Christina asks why it is better to work at Facebook than Google.

Use Shortest Paths between Entity Mentions

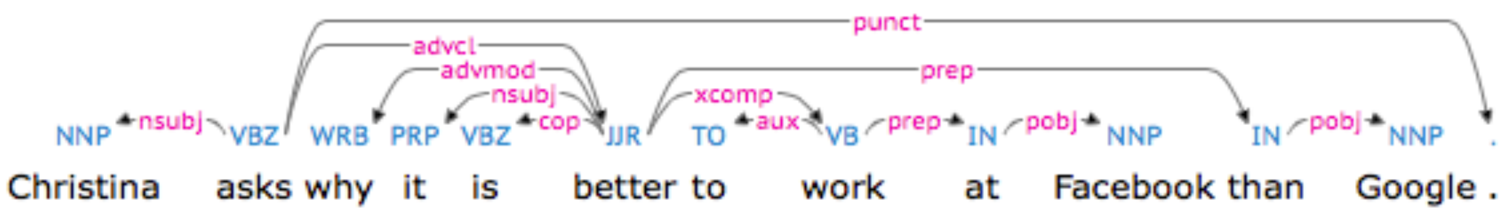


(Mark, Telstra) <-nsubj-prep->pobj->



(John, Facebook) <-nsubj-prep->pobj->

(Zuckerberg, Microsoft) <-nsubj-ccomp->pre



(Christina, Google) <-nsubj-advcl->prep->pobj->

Dependency Grammar

- Syntactic structure consists of lexical items, linked by binary asymmetric relations called dependencies.

- head ⟶ dep
  - head (governor) : grammatic [illegible] portant.
  - dependent (modifier): modifier, object, or complement.

Dependency Trees

- Without labels.

- With labels.

Dependency Parsing

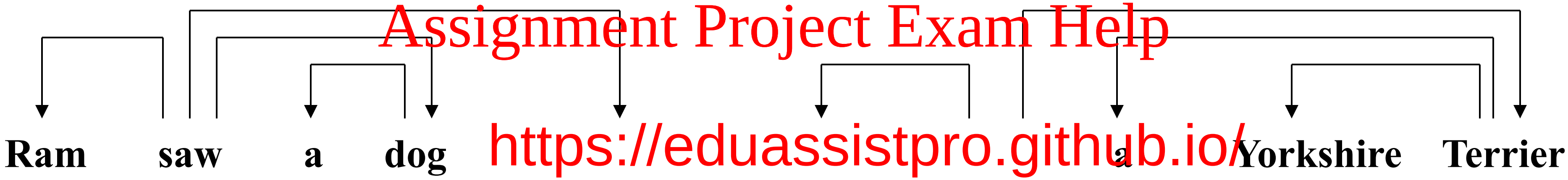
- Formal definition for unlabeled dependency trees:

Dependency graph $D = (V, E)$ where

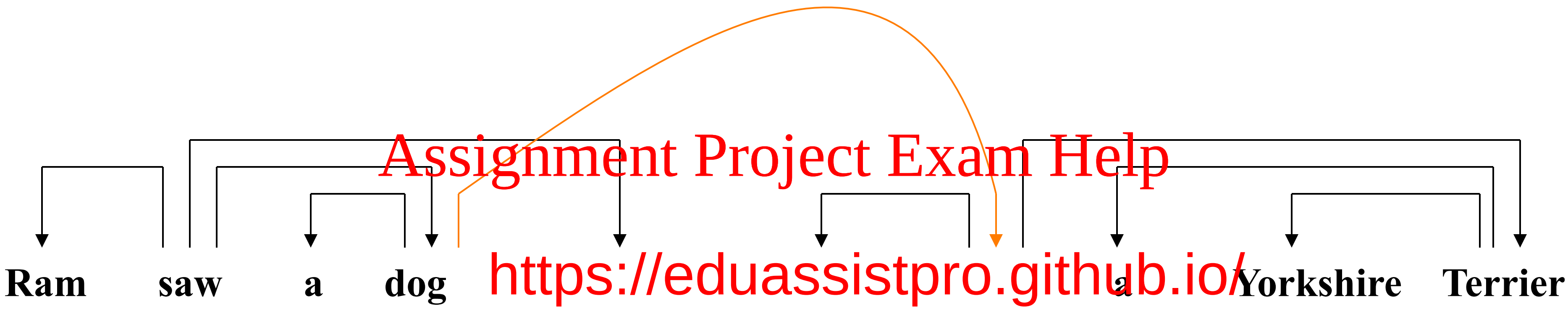
- V is the set of nodes (words in a input sequence).
- E is the set of arcs indi
- $v_i \to v_j$ or $(v_i, v_j) \in E$ denotes an ar v_i to dependent v_j.

- Dependency parsing: task of mapping an input string to a dependency graph satisfying certain conditions.

Projective Dependency Tree



Non-Projective Dependency Tree



Crossing lines!

English has very few non-projective cases.

Well-Formedness

- A dependency graph is well-formed iff
  - Single head: Each word has only one head.
  - Acyclic: The graph should be acyclic.
  - Connected: There is a path pairs of nodes.
  - Projective: iif an edge from word A to word B implies that there exists a directed path in the graph from A to every word between A and B in the sentences.

Parsing Algorithms

- Graph-based parsing
  - CYK, Eisner, McDonald

- Transition-based parsing
  - Covington, Yamada and Mat re etc.

Nivre's Algorithm (Arc-eager) [3]

- Transition-based.

- Parser configuration $< S, I, A >$:

  - S is the stack. $S[0]$ is the topmost node.
  - I is the list of remaini e leftmost word.
  - A is the set of current dependencies (arcs) f y graph.

- INPUT: a word sequence $\mathbf{v} = v_1|...|v_n$, a set of rules R.

Parser Transitions

Left-Arc (LA) $< v_i|S, v_j|I, A > \Rightarrow < S, v_j|I, A \cup \{(v_j, v_i)\}$ $\quad v_i \leftarrow v_j \in R$, $\nexists v_k (v_k, v_i) \in A$

single head

Right-Arc (RA) $< v_i|S, v_j|I, A > \Rightarrow < v_j|v_i|S, I, A \cup \{(v_i, v_j)\}$ $\quad v_i \rightarrow v_j \in R$, $\nexists v_k (v_k, v_j) \in A$

Reduce (R) $< v_i|S, I, A > \Rightarrow < S, I, A >$ $\quad \exists (v_k, v_i) \in A$

Shift (S) $< S, v_j|I, A > \Rightarrow < v_j|S, I, A >$

Parsing Details

- Slight modifications:
  - Each dependency graph has an artificial root in order to form a tree.
  - Parsing starts with an initial configuration $< [ROOT], \mathbf{n}, \emptyset >$ and terminates when it reaches $< [ROOT], \mathbf{nil}, A >$ through a sequence of tr

- Nondeterministic transitions?
  - Priority ordering of transition

    $\mathbf{LA} > \mathbf{RA} >$

    if $S[0]$ can be a transitive head of $I[0]$, then **Shift**, otherwise **Reduce**.
  - Guided parsing.

Grammatical Rules for the Example

$$Noun \rightarrow Adj$$

$$ROOT \rightarrow Verb$$

$$Noun \rightarrow Det$$

$$Verb \rightarrow N$$

$$figure \rightarrow on$$

$$on \rightarrow screen$$

Example

ROOT

ROOT [Red figure d falling stocks]

I

Example

red

ROOT

ROOT Red [figure falling stocks]

I

Shift

Example

ROOT

ROOT Red [figure falling stocks]

I

Left-arc

Example

figures

ROOT

ROOT Red figures [d falling stocks]

I

Shift

Example

on
figures

ROOT

ROOT Red figures [falling stocks]

I

Right-arc

Example

the
on
figures

ROOT

ROOT Red figures d falling stocks

I

Shift

Example

on
figures

ROOT

ROOT Red figures [illegible] d falling stocks

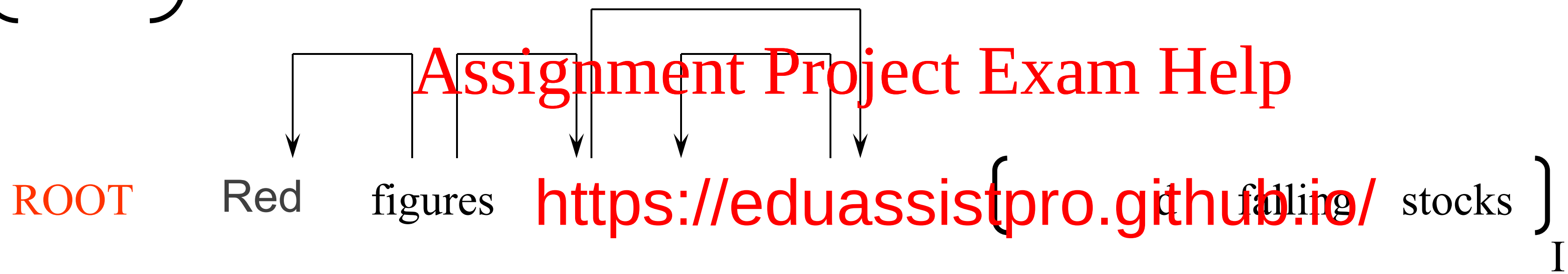
I

Left-arc

Example

screen
on
figures

ROOT



Right-arc

Example

on
figures

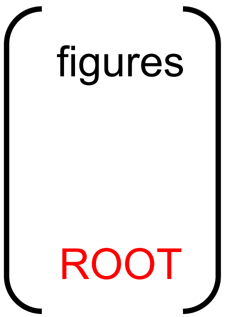
ROOT

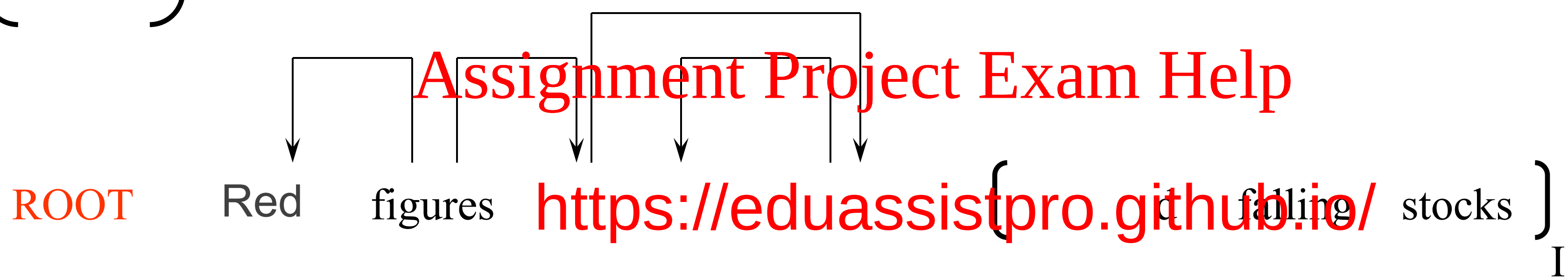
ROOT Red figures d falling stocks

I

Reduce

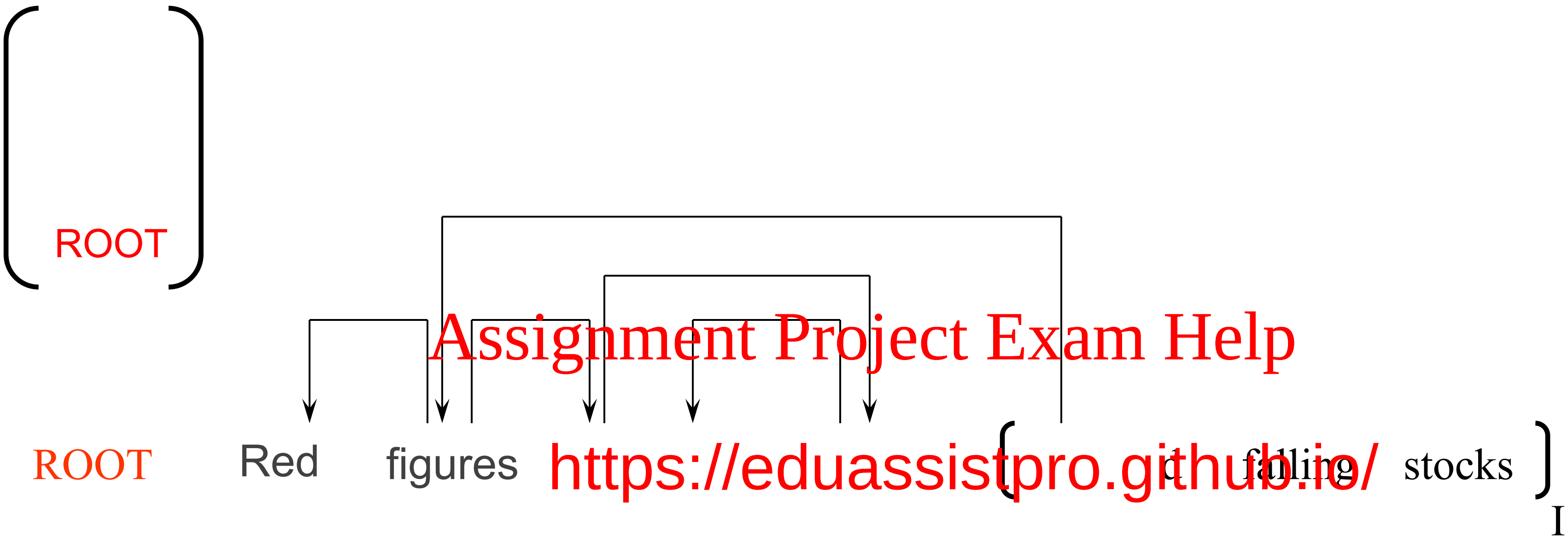
Example





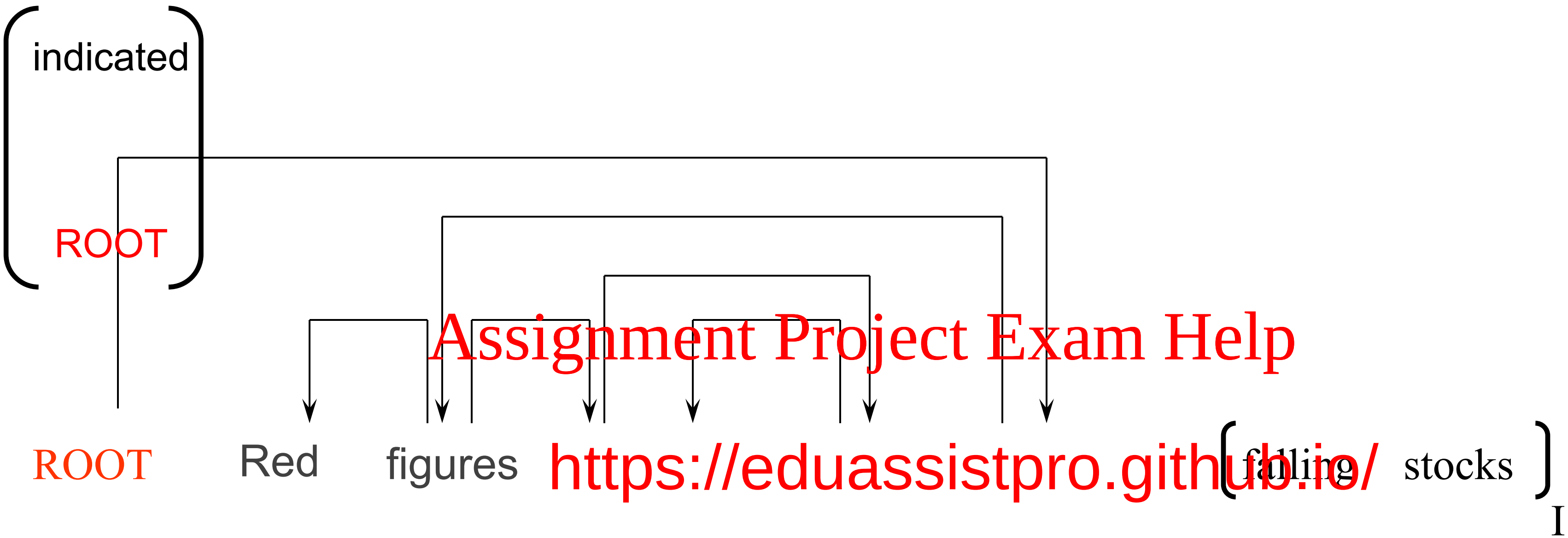
Reduce

Example



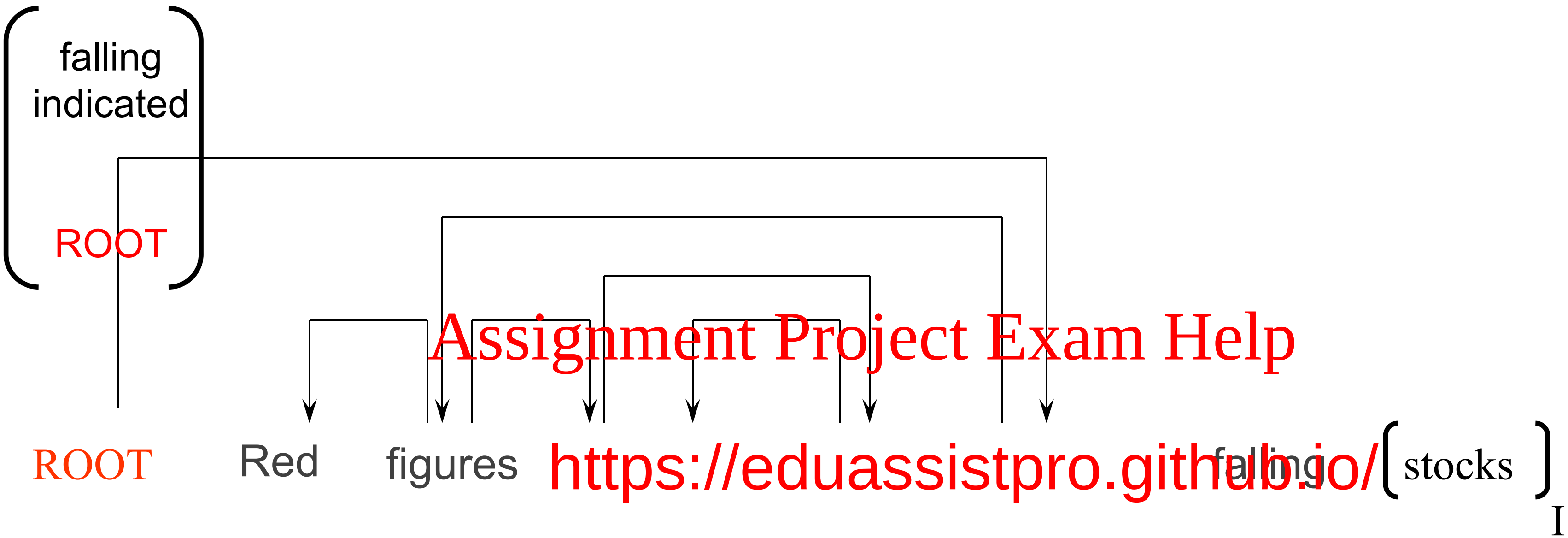
Left-arc

Example



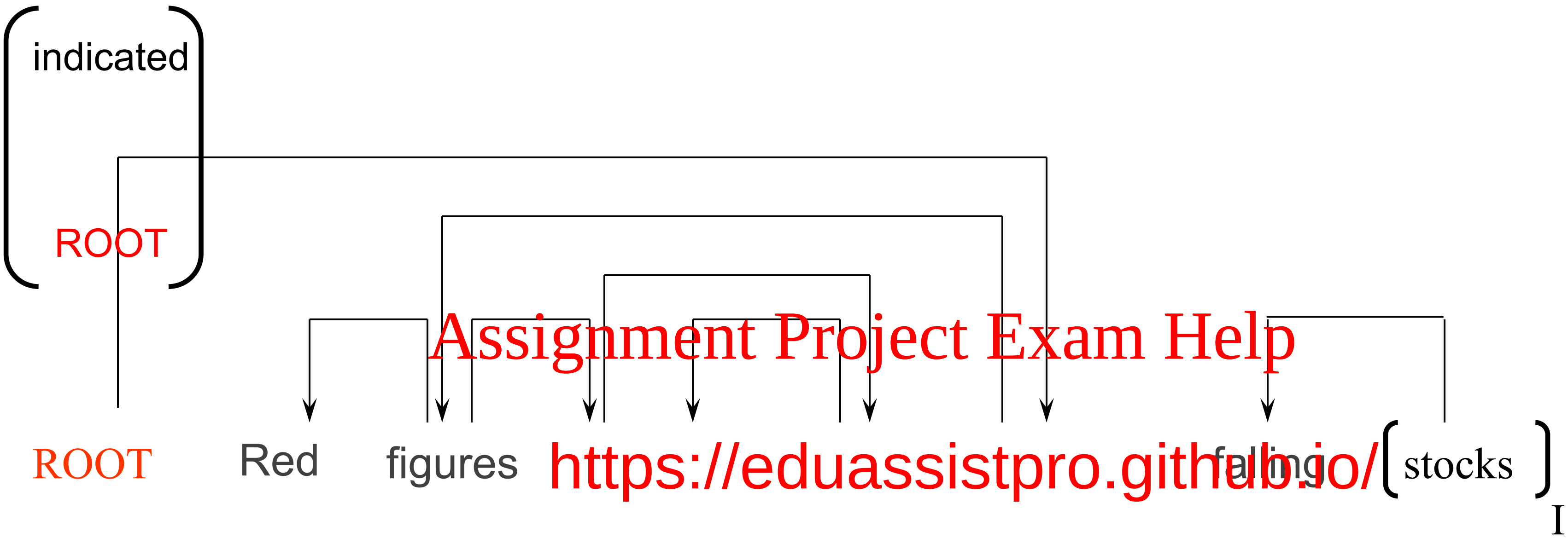
Right-arc

Example



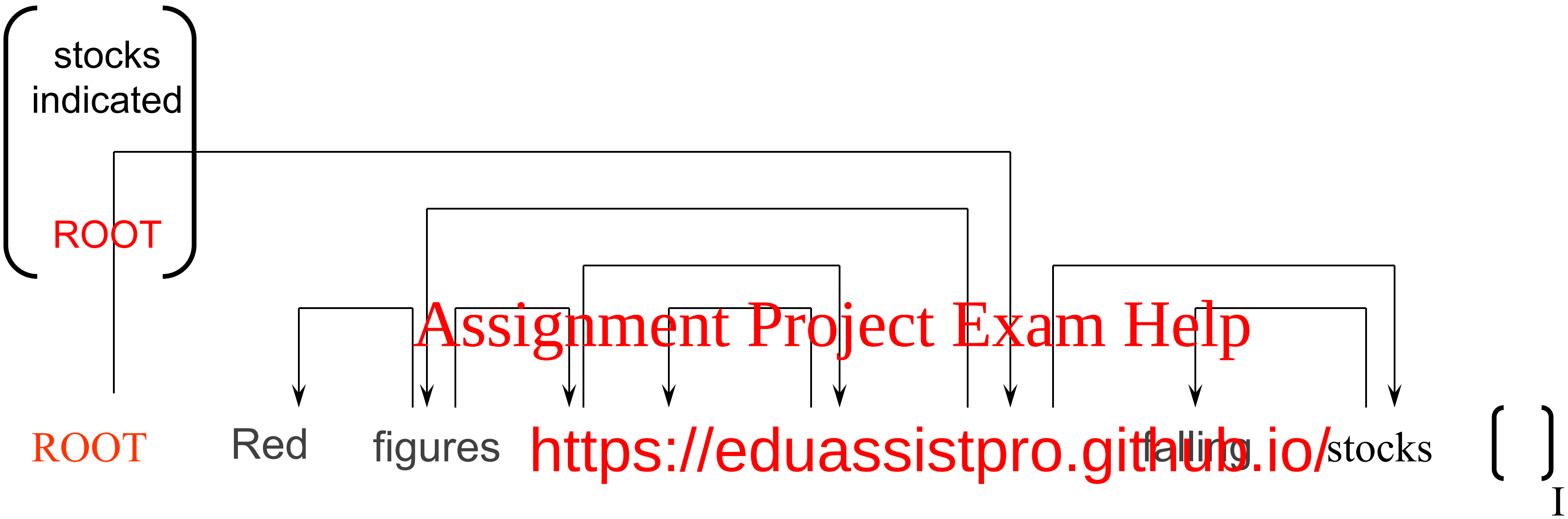
Shift

Example



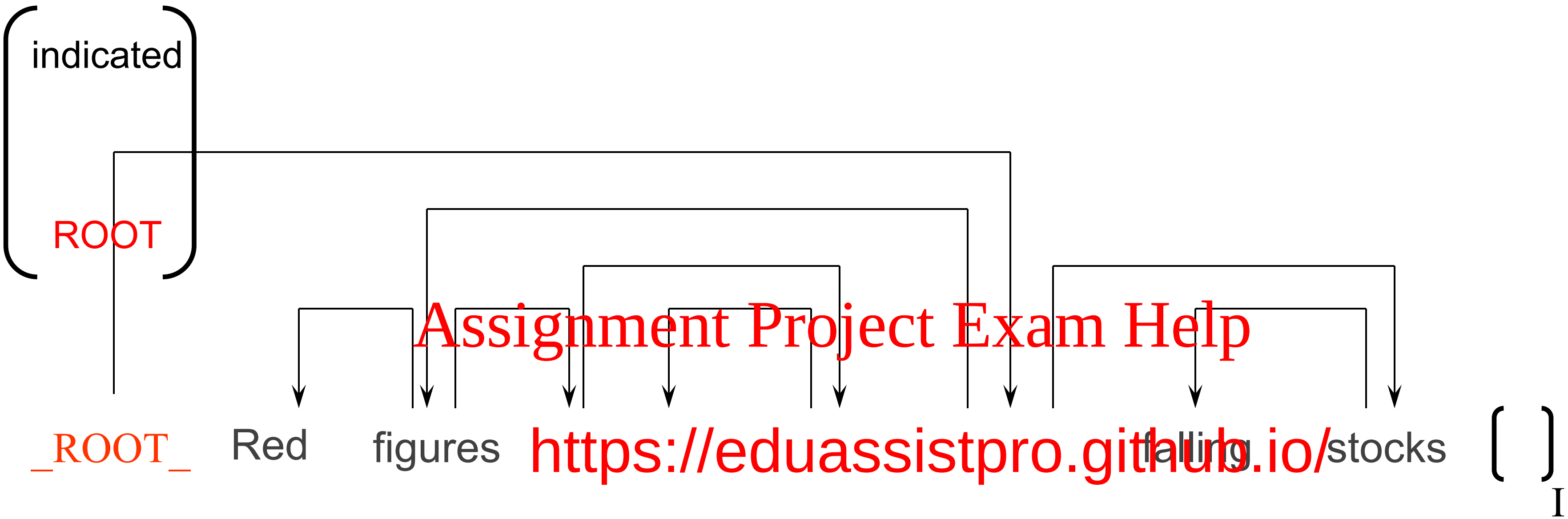
Left-arc

Example



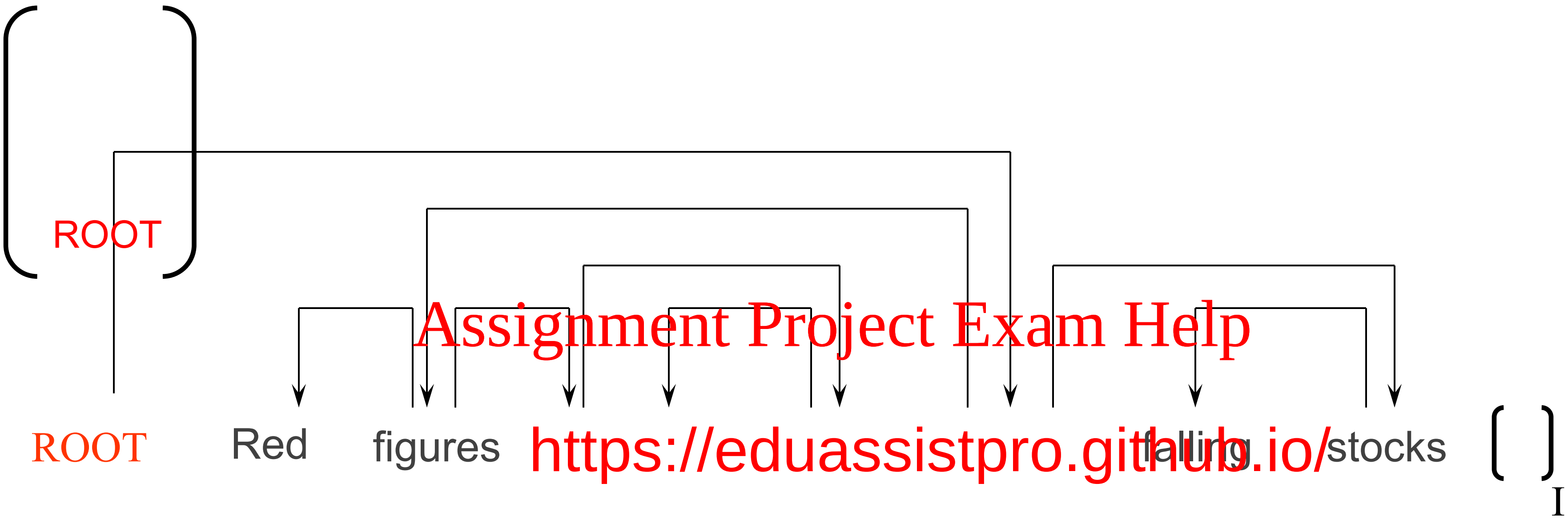
Right-arc

Example



Reduce

Example



Reduce

Configurations of the Example

<ROOT, Red figures on the screen indicated falling stocks, $\emptyset$>

S <Red ROOT, figures on the screen indicated falling stocks, $\emptyset$ >

LA <ROOT, figures on the screen indicated falling stocks, {(figures, Red)}>

S <figures ROOT, on the screen indicated falling stocks, {(figures, Red)}>

RA <on figures ROOT, the screen indicated falling stocks, {(figures, Red), (figures, on)}>

S <the on figures ROOT, screen indicated falling stocks, {(figures, Red), (figures, on)}>

LA <on figures ROOT, screen indicated falling stocks, {(figures, Red), (figures, on), (screen the)}>

RA <screen on figures ROOT, indicated falling stocks, {(figures, Red), (figures, on), (screen, the), (on, screen)}>

R <on figures ROOT, indic es, on), (screen, the), (on, screen)})

R <figures ROOT, indicate on), (screen, the), (on, screen)}>

LA <ROOT, indicated falling creen, the), (on, screen), (indicated, figures)}>

RA <indicated ROOT, falling stocks, {(figures, Red), (n, the), (on, screen), (indicated, figures), (ROOT, indicated)}>

S <falling indicated ROOT, stocks, {(figures, Red), (figures, on), (screen, the), (on, screen), (indicated, figures), (ROOT, indicated)}>

LA <indicated ROOT, stocks, {(figures, Red), (figures, on), (screen, the), (on, screen), (indicated, figures), (ROOT, indicated), (stocks, falling)}>

RA <stocks indicated ROOT, **nil**, {(figures, Red), (figures, on), (screen, the), (on, screen), (indicated, figures), (ROOT, indicated), (stocks, falling), (indicated, stocks)}>

R <indicated ROOT, **nil**, {(figures, Red), (figures, on), (screen, the), (on, screen), (indicated, figures), (ROOT, indicated), (stocks, falling), (indicated, stocks)}>

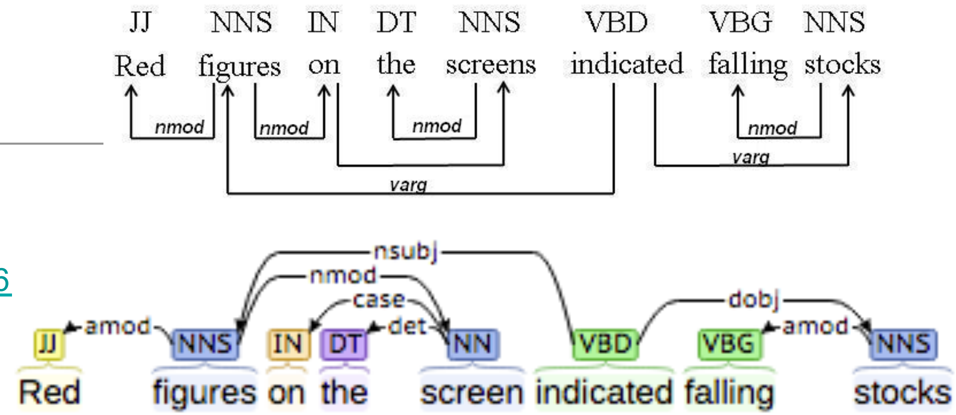
R <ROOT, **nil**, {(figures, Red), (figures, on), (screen, the), (on, screen), (indicated, figures), (ROOT, indicated), (stocks, falling), (indicated, stocks)}>

Properties of Nivre’s Algorithm

- O(n) : Linear time complexity.

- Full dependency graphs ar ed.

Dependency Corpora



- CoNLL dependencies.
  - http://www.aclweb.org/anthology/D07-1096
- Stanford typed dependencies.
  - http://nlp.stanford.edu/software/dependencies_manual.pdf
- Universal dep
  - http://universaldepende

Guided Parsing [6]

- Train a classifier to predict parse transitions!
  - $f : configuration \rightarrow \{LA, RA, R, S\} \times (A \cup \{\text{nil}\})$
  - A is a set of typed dependencies (arcs).

- Feature spac

Arc Standard

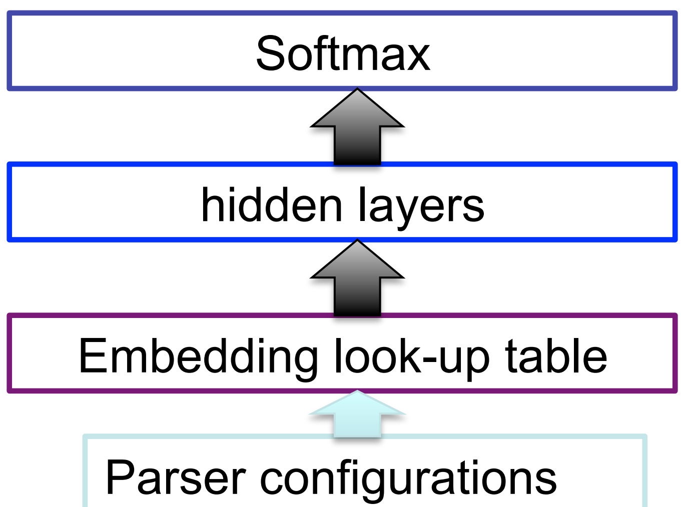
- Three parse actions.

Left-Arc (LA) $< v_i|v_j|S, I, A > \Rightarrow < v_i|S, I, A \cup \{(v_i, v_j)\}$

Right-Arc (RA) $< v_i|v_j|S, I, A > \Rightarrow < v_j|S, I, A \cup \{(v_j, v_i)\}$

Shift (S) $A > \Rightarrow < v_j|S, I, A >$

- Neural networks for action [9].



Off-the-Shelf Dependency Parsers

- MaltParser (http://www.maltparser.org/)
- SyntaxNet (https://github.com/tensorflow/models/tree/master/research/syntaxnet)
- Stanford parse arser.shtml)
- TurboParser (http://www.cs.cmu.edu/~ark/Tur

Overview of the NLP Lectures

- Introduction to natural language processing (NLP).
- Regular expressions, sentence splitting, tokenization, part-of-speech tagging.

- Language mod

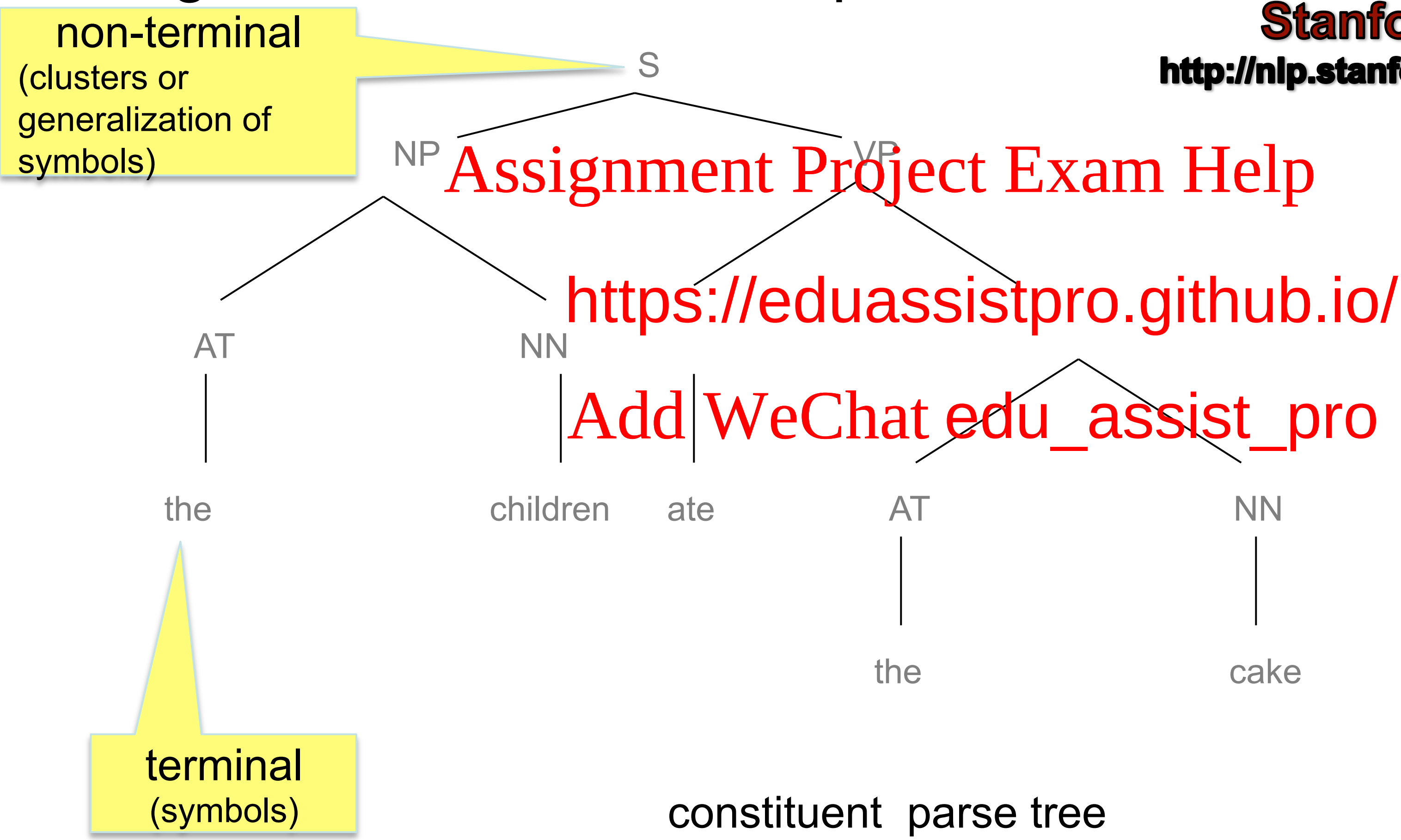
- Vector semantics.

- Parsing.
  - Dependency parsing.
  - Constituency parsing.
- Compositional semantics and NLP applications.

Constituency Parsing

- Deeper understanding of word groups and their grammatical relationships.

Stanford Parser
http://nlp.stanford.edu:8080/parser



constituent parse tree

Constituency

- Constituent: a word or a group of words that behaves as a single unit.

- Why do these words group together?
  - Appear in si ts.

three parties from Drunk driver *fled* … they *sit* …	from *arrive* … the *fled* … as *sit* …

  - Preposed or postposed construction.

On August 30th, I'd like to fly from Canberra to Sydney.
I'd like to fly on August 30th from Canberra to Sydney.
I'd like to fly from Canberra to Sydney on August 30th.

Context-Free Grammars (CFGs)

- A context free grammar consists of
  - a set of context-free rules, each of which expresses the ways that symbols of the language can be grouped and ordered together.

NP $\longrightarrow$ Det Nominal

Nomina $\longrightarrow$

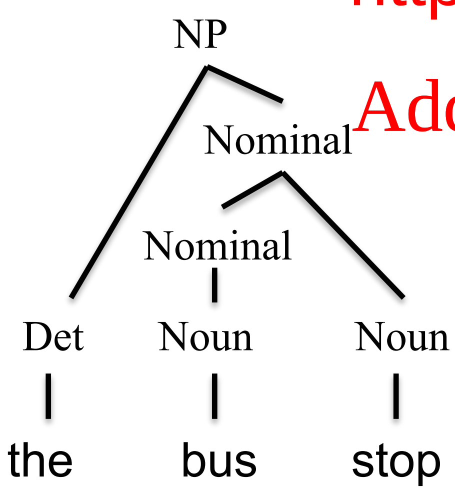
Nomina $\longrightarrow$

  - a lexicon of words and symb

bus stop the . a …

Derivations

- The sequence of rule expansions is called a derivation of the string of words.
  - parse tree.
  - bracketed notation.



[*NP* [*Det* the][*Nom* [*Nom* [*Noun* bus]][*Noun* stop]]

A Toy Example

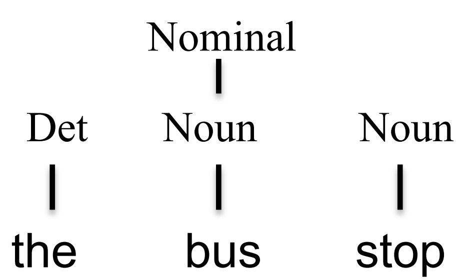
Noun	$\longrightarrow$	bus
Noun	$\longrightarrow$	stop
Det	$\longrightarrow$	the | a | an
Nominal	$\longrightarrow$	Noun
NP	$\longrightarrow$	Det Nominal
Nominal	$\longrightarrow$	Noun | Nominal Noun

the bus stop

A Toy Example

Noun → bus
Noun → stop
Det → the | a | an

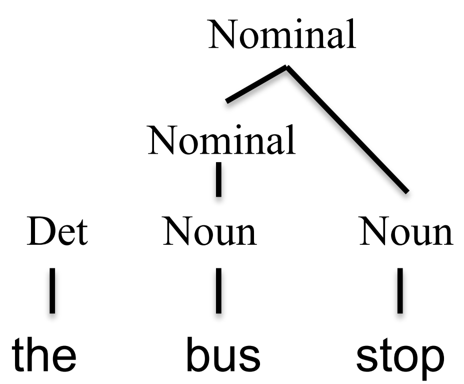
Nominal → Noun
NP → Det Nominal
Nominal → Noun | Nominal Noun



A Toy Example

Noun $\rightarrow$ bus
Noun $\rightarrow$ stop
Det $\rightarrow$ the | a | an

Nominal $\rightarrow$ Noun
NP $\rightarrow$ Det Nominal
Nominal $\rightarrow$ Noun | Nominal Noun



A Toy Example

Noun $\longrightarrow$ bus

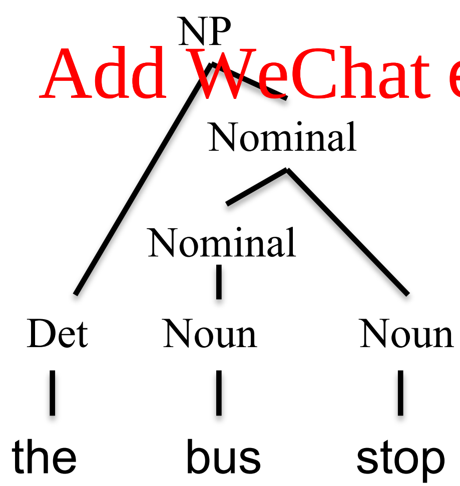
Noun $\longrightarrow$ stop

Det $\longrightarrow$ the | a | an

Nominal $\longrightarrow$ Noun

NP $\longrightarrow$ Det Nominal

Nominal $\longrightarrow$ Noun | Nominal Noun



Formal Definition of CFG

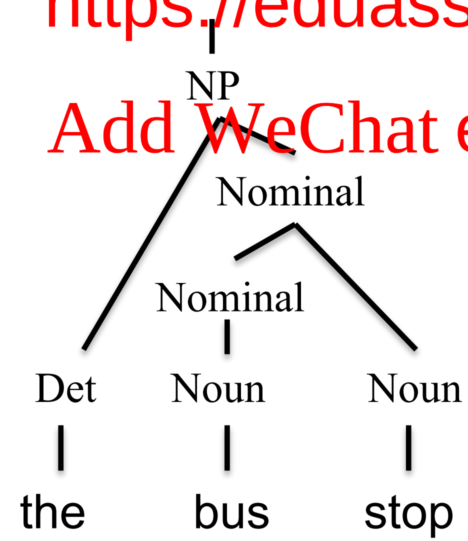
- A context-free grammar $G = (N, \Sigma, R, S)$.
  - N is a set of non-terminals.
  - Σ is a set of terminal symbols, $N \cap \Sigma = \emptyset$.

  - R is a set of rules (productions), each of the form $A \to B$, where A is a g of symbols from the infinite se

  - S is a designated start symb

A Toy Example

Noun → bus
Noun → stop
Det → the | a | an
S → NP
Nominal → Noun
NP → Det Nominal
Nominal → Noun | Nominal Noun



Ambiguity of Parsing

Probabilistic context-free grammar (PCFG)

- A parameter to each grammar rule [3].

$$p_G(t) = \prod_{i=1}^{n} q(\alpha \to \beta)$$

rule parameter

$$\arg\max_{t \in T_G} p_G(t)$$

find the most likely parse tree. T is set of all possible trees.

Learning PCFG from Treebanks

- Penn treebank and English Web treebank.

Maximum-Likelihood estimation: $$q^*(\alpha \rightarrow \beta) = \frac{\mathrm{Count}(\alpha \rightarrow \beta)}{\mathrm{Count}(\alpha)}$$

Top Down Parsing $\arg\max_{t \in T_G} p_G(t)$

book **that** **flight**

Bottom Up Parsing $\arg\max_{t \in T_G} p_G(t)$

Grammar Equivalence

- Two grammars are equivalent if they generate the same language (set of strings) .
- Chomsky Normal Form (CNF).
  - Allow only two types of rules. The right-hand side of each rule either has two non-terminals or one terminal,

except $S \to \varepsilon$

$$A \to B\ a\ D$$

$$C \to \varepsilon$$

unit production

$$E \to A$$

where $A, B, C, D, E \in N$ and $a \in \Sigma$

Grammar Equivalence

- Two grammars are equivalent if they generate the same language (set of strings) .
- Chomsky Normal Form (CNF).
  - Allow only two types of rules. The right-hand side of each rule either has two non-terminals or one terminal,

    except $S \rightarrow \varepsilon$

$$
\begin{aligned}
&G \rightarrow E\ D \\
&F \rightarrow B\ G \\
&E \rightarrow a \\
&\text{where } B, D, E, F, G \in N \text{ and } a \in \Sigma
\end{aligned}
$$

  - Every context-free grammar can be transformed into an equivalent one in CNF.

Dependency Structures vs. Phrase Structures

- Dependency structures explicitly represent
  - Head-dependent relations (directed arcs).
  - Functional categories (arc labels).
  - predicate-argument structure.

- Dependency of word order.
  - Suitable for f such as Indian languages.

- Phrase structures explicitly represent
  - Phrases (non-terminal nodes).
  - Structural categories (non-terminal labels).
  - Fragments are directly interpretable.

Available Constituency Parsers

- Stanford parser.
  - http://nlp.stanford.edu/software/srparser.shtml

- Charniak-Johnson parser.
  - http://web.science.mq.edu.au/~mjohnson/Software.htm

- Charniak par
  - ftp://ftp.cs.brown.edu/pub/nlp

References

- [1] Speech and Language Processing (Chapter 12 & 13)
- [2] Lectures Notes on CFGs and CKY Parsing.
  - http://web.mit.edu/6.863/www/fall2012/lectures/lecture7-notes.pdf
- [3] http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/pcfgs.pdf
- [4] Richard Socher, Christopher D. Manning, Andrew Y. Ng. Learning Continuous Phrase Representations and Syntactic Parsing with Recursive Neural Networks.
- [5] An Efficient Algo yParsing (We modify it to allow an artificial
- [6] Joakim Nivre, Johan Hall, Jens Nils ased Dependency Parsing.
- [7] Exploiting Background Knowledge for Relation Extraction.
  - http://www.aclweb.org/anthology/C10-1018
- [8] https://web.stanford.edu/~jurafsky/slp3/14.pdf
- [9] https://cs.stanford.edu/~danqi/papers/emnlp2014.pdf